UNITED STATES PATENT APPLICATION

For

EXPRESSION EDITOR

Inventor:

Hugh S. Njemanze

Prepared by:

Blakely, Sokoloff, Taylor & Zafman
12400 Wilshire Boulevard
Seventh Floor
Los Angeles, California 90025
(408) 947-8200

Attorney's Docket No. 6388P012

"Express Mail" mailing label number: EV301790348US
Date of Deposit: October 30, 2003
I hereby certify that I am causing this paper or fee to be deposited with the United States Postal Service "Express Mail Post Office to Addressee" service on the date indicated above and that this paper or fee has been addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450.

Vineta T. Tufono
(Typed or printed name of person mailing paper or fee)

Vineta T. Tufono
(Signature of person mailing paper or fee)

10/30/2003
(Date signed)

EXPRESSION EDITOR

FIELD OF THE INVENTION

[0001] The present invention relates to expressions, and, in particular, to displaying and editing these expressions.

BACKGROUND

[0002] Various expressions, such as Boolean and arithmetic expressions, are used in computer technology. Many search engines, for example, use Boolean expressions as search terms. More lengthy and complex rules, for example those used by network security systems, are also commonly represented and created using Boolean and other expressions.

[0003] Visual tree representations are also common in computer technology. For example, directory trees are used as graphical user interfaces to display the organizational hierarchy of a directory. Expressions can also be displayed according to such a tree structure. Such a tree structure is sometimes referred to as an expression tree. Expression trees display expressions in prefix notation. Because of their inherent prefix nature, expression trees can be difficult for a human to read and edit.

SUMMARY OF THE INVENTION

[0004] A prefix expression tree showing an expression can be supplemented to also display the expression in infix notation. In one embodiment, the present invention includes displaying an expression being capable of representation in infix and prefix notation in prefix expression tree format. In one embodiment, the expression includes a plurality of operators and operands, and the plurality of operands make up the leaves of the expression tree. In one embodiment, the present invention further includes inserting a plurality of infix operators corresponding with the plurality of operators into the prefix expression tree, wherein, the plurality of operands and infix operators represent the expression in infix notation.

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] The present invention is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings in which like reference numerals refer to similar elements and in which:

[0006] **Figure 1** is a block diagram illustrating a directory tree; and

[0007] **Figure 2** is a block diagram illustrating a Boolean expression tree;

[0008] **Figure 3** is a block diagram illustrating a hybrid expression tree according to one embodiment of the present invention;

[0009] **Figure 4** is a block diagram illustrating a network security system in which embodiments of the present invention may be implemented;

[0010] **Figure 5** is a block diagram illustrating a rule viewed as an expression tree according to one embodiment of the present invention;

[0011] **Figure 6** is a block diagram illustrating a rule viewed as a hybrid expression tree according to one embodiment of the present invention; and

[0012] **Figure 7** is a screenshot of a rule viewer/editor according to one embodiment of the present invention.

DETAILED DESCRIPTION

[0013] Described herein is an expression editor that can represent an expression in prefix tree notation and infix notation simultaneously.

[0014] Although the present system will be discussed with reference to various illustrated examples, these examples should not be read to limit the broader spirit and scope of the present invention. For example, the examples presented herein describe distributed agents, managers and various network devices, which are but one embodiment of the present invention. The general concepts and reach of the present invention are much broader and may extend to any computer-based or network-based security system.

[0015] Some portions of the detailed description that follows are presented in terms of algorithms and symbolic representations of operations on data within a computer memory. These algorithmic descriptions and representations are the means used by those skilled in the computer science arts to most effectively convey the substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of steps leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared and otherwise manipulated. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers or the like. It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise, it will be appreciated that throughout the description of the

present invention, use of terms such as "processing", "computing", "calculating", "determining", "displaying" or the like, refer to the action and processes of a computer system, or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

[0016] As indicated above, one embodiment of the present invention is instantiated in computer software, that is, computer readable instructions, which, when executed by one or more computer processors/systems, instruct the processors/systems to perform the designated actions. Such computer software may be resident in one or more computer readable media, such as hard drives, CD-ROMs, DVD-ROMs, read-only memory, read-write memory and so on. Such software may be distributed on one or more of these media, or may be made available for download across one or more computer networks (e.g., the Internet). Regardless of the format, the computer programming, rendering and processing techniques discussed herein are simply examples of the types of programming, rendering and processing techniques that may be used to implement aspects of the present invention. These examples should in no way limit the present invention, which is best understood with reference to the claims that follow this description.

Boolean Expressions and Boolean Expression Trees

[0017] A Boolean expression is a statement using Boolean operators that expresses a condition that is either true or false. Some well known Boolean operators are AND, OR, and NOT. A Boolean expression thus contains operands and Boolean operators. Operands are conditions that are either true or false. For example, if the operand **a** is TRUE and the operand **b** is FALSE, then the expression (**a** AND **b**) is FALSE, since **a** and **b** are not both TRUE.

[0018] Boolean expressions are well known to persons skilled in the art. Many laypeople also encounter Boolean expressions while using Internet search engines such as Google™ and LEXIS/NEXIS™. For example the Boolean search expression (ricky AND martin AND NOT rowan) finds documents presumably related to singer Ricky Martin and unrelated to Rowan and Martin's Laugh-In.

[0019] The sample Boolean expressions above are all given in infix notation. In infix notation, the operator appears between the operands. For example, in the expression ((**a** AND **b**) OR **c**), the operator AND is between operands **a** and **b**, and operator OR is between operands **(a AND b)** and **c**. The parentheses indicate the order of the operations. Humans find it easy to read infix notation, which is why most mathematical expressions are expressed using such notation, e.g., 1+2.

[0020] However, machines generally use prefix notation. In prefix notation, the operator comes first, followed by the operands, e.g. +,1,2. The above expression ((**a** AND **b**) OR **c**) in prefix notation would be OR AND **a b c**. One advantage of prefix notation is that the order of operations is implicit, and parentheses are not required. In

postfix notation, sometimes referred to as reverse polish notation, the operands come before the operator.

[0021] There are other expressions besides Boolean expressions that contain operators and operands, which similarly have both infix and prefix notational representations. Arithmetic expressions, for example, are not Boolean expression. They do, however, have both infix and prefix notational representations, as shown above with reference to the arithmetic expression (1+2). The present invention encompasses all expressions, both Boolean and not, that can be represented in both a prefix and an infix notation. However, for simplicity and ease of understanding, most examples described herein use Boolean expressions. Applying the principles demonstrated by these examples to non-Boolean expressions that are capable of both infix and prefix representation would be obvious to one skilled in the art.

[0022] Most people encounter tree representations in everyday computing in the form of a directory tree. Figure 1 shows a typical directory tree displaying the contents of the Files Folder 100. The Files Folder 100 contains Folders 102-105. Folder 102 contains Document 108, Folder 105 contains Documents 110-111, and so on. Expansion buttons 112-113 indicate whether the contents of a folder are displayed, and can be used to save space on a display screen. The Files Folder 100 is referred to as the root of the tree, and the documents are referred to as the leaves, since they contain no additional files.

[0023] Boolean expressions can be represented in prefix notation using a similar tree structure. For example, Figure 2 shows such a Boolean expression tree for the

expression ((**a** AND **b**) OR **c**). The Boolean expression tree is a prefix tree. That is, reading top-to-bottom, the operators 202 appear before the operands 204.

[0024] Partly due to its prefix nature, such Boolean expression trees can be difficult for humans to read. Even such a simple expression as the one shown in Figure 2 requires five directional reversals during reading. More complex expressions become impossible for humans to read and edit without first writing them out in infix notation.

Hybrid Infix/Prefix Boolean Expression Tree

[0025] In one embodiment, the prefix Boolean expression tree is augmented with some infix notation to make it more readable. One embodiment of how this can be done is demonstrated with reference to Figure 3, which shows the same expression as Figure 2. The operators 202 and the horizontal lines of the tree appear to be similar to Figure 2. In one embodiment, the operands 204 are moved to the right of an invisible line, so that the operands 204 can be read together without reference to the tree. However, the operands 204 are still displayed on the proper connecting lines of the tree, and can still be read as part of the tree.

[0026] In one embodiment, infix operators 300 are added to the operand side of the tree, that is, on the right of the invisible line described above and shown in Figure 3. The infix operators 300 AND and OR are inserted between the operands 204 such that when reading on the right side of the invisible line (the operand side of the Boolean expression tree) from top-to-bottom and left-to-right, the expression reads as (**a** AND **b**) OR **c**. Thus, the right hand side of the prefix tree can now also be read in infix notation.

[0027] In one embodiment, the method for inserting the infix operators 300 is to copy the applicable operator 202 and insert it between the operands 204 it operates on. In Figure 3, for example, AND is inserted between a and b. Also, OR is inserted between (a AND b) and c. In the embodiment demonstrated with reference to Figure 3, the operands 204 have remained on the line that indicates their position on the prefix Boolean expression tree, and the insertion of the infix operators 300 was done in a manner that preserves these positions. Furthermore, in the embodiment demonstrated with reference to Figure 3, the operands 204 all appear on the right of an invisible line dividing the Boolean expression tree form the operands 204. This can make the infix operand side of the expression tree easier to read once the infix operators 300 have been inserted.

Example Rules Editor

[0028] One embodiment of the present invention is now demonstrated in the context of a computer security system. A computer security system can analyze events and alarms from various network security products. Network security products largely include Intrusion Detection Systems (IDSs), which can be Network or Host based (NIDS and HIDS respectively). Other network security products include firewalls, router logs, and various other event reporting devices. An example network security system is now described with reference to Figure 4.

[0029] In Figure 4, an example of a computer-based system 400 architected in accordance with an embodiment of the present invention is illustrated. System 400 includes agents 412, one or more managers 414 and one or more consoles 416 (which may include browser-based versions thereof). In some embodiments, agents, managers

and/or consoles may be combined in a single platform or distributed in two, three or more platforms (such as in the illustrated example). The use of this multi-tier architecture supports scalability as a computer network or system grows.

[0030] Agents 412 are software programs that provide efficient, real-time (or near real-time) local event data capture and filtering from a variety of network security devices and/or applications. The primary sources of security events are common network elements including firewalls, intrusion detection systems and operating system logs. Agents 412 can collect events from any source that produces event logs or messages and can operate at the native device, at consolidation points within the network, and/or through simple network management protocol (SNMP) traps.

[0031] Managers 414 are server-based components that further consolidate; filter and cross-correlate events received from the agents, employing a rules engine 418 and a centralized event database 420. One role of manager 414 is to capture and store all of the real-time and historic event data to construct (via database manager 422) a complete, enterprise-wide picture of security activity. The manager 414 also provides centralized administration, notification (through one or more notifiers 424), and reporting, as well as a knowledge base 428 and case management workflow. The manager 414 may be deployed on any computer hardware platform and one embodiment utilizes a relational database management system such as an Oracle™ database to implement the event data store component. Communications between manager 414 and agents 412 may be bi-directional (e.g., to allow manager 414 to transmit commands to the platforms hosting agents 412) and encrypted.

[0032] Consoles 416 are computer- (e.g., workstation-) based applications that allow security professionals to perform day-to-day administrative and operation tasks such as event monitoring, rules authoring, incident investigation and reporting. Access control lists allow multiple security professionals to use the same system and event database, with each having their own views, correlation rules, alerts, reports and knowledge base appropriate to their responsibilities. A single manager 414 can support multiple consoles 416.

[0033] In some embodiments, a browser-based version of the console 416 may be used to provide access to security events, knowledge base articles, reports, notifications and cases. That is, the manager 414 may include a web server component accessible via a web browser hosted on a personal or handheld computer (which takes the place of console 416) to provide some or all of the functionality of a console 416. Browser access is particularly useful for security professionals that are away from the consoles 416 and for part-time users. Communication between consoles 416 and manager 412 is bi-directional and may be encrypted.

[0034] As mentioned above, the console interface 416 can be used to author rules for the rules engine 418 or the manager 414. In one embodiment, these rules take the form of Boolean expressions. For example a rule representing a successful brute force attack can be ((ten unsuccessful logon attempts) AND (successful logon from same IP address)).

[0035] The consoles 416 can similarly be used to author and edit other Boolean expressions, such as rules that control filters. In one embodiment, the rule viewer/editor that creates the user interface to view and edit the rules on the consoles 416 uses a hybrid

prefix/infix Boolean expression tree of the variety described with reference to Figure 3. An example rule as displayed by such a display/editor is described with reference to Figure 6. To contrast, a the example rule is first shown as a pure prefix expression tree in Figure 5.

[0036] The rule displayed in Figures 5 and 6 is the definition of "State," a sub-rule in the rule "Matching Event," that can be used to detect matching events. The entire rule can be seen on the sample rule editor shown in the screenshot in Figure 7. The State is defined by a Boolean expression so that State is TRUE if the event identifier identifies the event as the Router-Interface Changed State (Event ID = Router-Interface Changed State), the event is a correlated type event (Event Type = Correlated), and, according to the proprietary ArcSight™ classification, the event is in the correlated category (ArcSight Category = Correlated). In the alternative, State is also TRUE if the name of the event is "Router-Line State Changed" (Event Name = Router-Line State Changed), the event is a correlated type event (Event Type = Correlated), and the event is in the correlated category (ArcSight Category = Correlated). If neither of the previous two sentences is TRUE, then the State is FALSE.

[0037] In infix notation State is: ((Event ID = Router-Interface Changed State AND Event Type = Correlated AND ArcSight Category = Correlated) OR (Event Name = Router-Line State Changed AND Event Type = Correlated AND ArcSight Category = Correlated)). Figure 5 shows this expression as it appears using a prefix Boolean expression tree arranged in a visually pleasing format. In this embodiment, the operators are represented in graphical form. That is, operator 502 stands for OR – in the commonly used graphical representation as | | - and operators 504 stand for AND – in the commonly

used graphical representation as &. In other embodiments, these operators can be in a language format, such as English. The expansion boxes 506 can be used to simplify the tree by hiding the operands 508 of the operators 504.

[0038] The operands 508 are aligned to the left on the right side of the invisible line 510 discussed above. An operand, such as Event Type = correlated is TRUE or FALSE depending on whether the stated condition is true or false. Such a graphical representation can be very useful and convenient. However, it is difficult to read the rule out loud without first writing it in infix notation. More complicated and lengthy rules become almost impossible to decipher by a human looking at the prefix Boolean expression tree.

[0039] Figure 6 shows how the above rule can be converted into infix notation, while keeping it in the context of the prefix Boolean expression tree. Since Figure 6 contains many of the same visual elements as Figure 5, those elements already labeled in Figure 5 are not labeled and described again for clarity, visual simplicity, and ease of understanding. In Figure 6, on the left hand of the invisible line 510 is the prefix tree side 602 of the Boolean expression tree, which is identical to the left side of the tree in Figure 5.

[0040] However, to the right of the invisible line 520 is now the infix/prefix hybrid side 604 of the Boolean expression tree. On this side, the operands 508 are still part of the Boolean expression tree. The operands are still on their appropriate vertical lines, such that they remain connected to the prefix Boolean expression tree. However, infix operators 606 have been inserted, so that when read from left-to-right and top-to-bottom (in the manner readers of English and many other languages commonly read), the

expression reads as an infix notation expression. In this embodiment, the infix operators 602 are inserted textually in English/language representation as opposed to symbolically in graphical notation; for example AND instead of &, but in other embodiments graphical equivalents or other languages can be used.

[0041] In one embodiment, the method for inserting the infix operators 606 can be established according to the following rules. Each operator on the prefix tree side 602 has an open parenthesis on the same line on the infix/prefix hybrid side 604. Furthermore, excluding the first operand of each operator, every operand has the operator inserted as an infix operator 606 in front of the operand. For example, Event Type = Correlated is the third operand of the first &. Thus, the infix operator AND is inserted in front of it on the infix/prefix hybrid side 604. Similarly, (Event Name = Router-Line State Changed AND Event Type = Correlated AND ArcSight Category = Correlated) is the second operand of the | | operator, thus, the infix operator OR is inserted in front of it. Close parentheses are inserted at the end of each operand list to coincide with the open parentheses.

[0042] It can be seen with reference to Figure 6, that following the above rules results in the infix operators 606 always being inserted between the operands when read in the usual left-to-right and top-to-bottom manner. Furthermore, the parentheses are also always properly inserted to indicate the order of the operations implicit in the Boolean expression tree. Furthermore, the operands 508 remain part of the prefix tree, while also readable in infix notation.

[0043] Figure 7 shows one embodiment of how a rule editor can use the infix/prefix hybrid Boolean expression tree described above. Figure 7 is a screenshot

from such a rule editor that appears in the Inspect/Edit window. Figure 7 displays the entire Matching Event rule, a part of which was used above. In one embodiment, if a user wants to edit a rule, then he or she can do so from either the graphical prefix tree side, or from the English language hybrid prefix/infix side. In other words, if on the hybrid side an AND is changed to OR for example, the graphical Boolean expression tree is updated and restructured to reflect the change. Conversely, if the | | is changed to & for example, the hybrid prefix/infix side is also updated to reflect the change.

[0044] Thus, a prefix/infix hybrid expression tree, and a rule inspector/editor that uses such a structure have been described. In the foregoing description various specific modules, such as the "Consoles," have been described. However, these names are merely to describe and illustrate various aspects of the present invention, and in no way limit the scope of the present invention. Furthermore, the present invention is not limited to use as a rule editor in a network security system, but can be used to view and edit any expression that has both an infix and a prefix representation.

[0045] The present invention is not limited to displaying and editing Boolean expressions. In the foregoing description, the various examples and embodiments were meant to be illustrative of the present invention and not restrictive in terms of heir scope. Accordingly, the invention should be measured only in terms of the claims, which follow.